

# Inactivation Decoding Analysis for LT Codes

Francisco Lázaro, Gianluigi Liva  
Institute of Communications and Navigation  
DLR (German Aerospace Center), Wessling, Germany  
Email: {Francisco.LazaroBlasco,Gianluigi.Liva}@dlr.de

Gerhard Bauch  
Institute for Telecommunication  
Hamburg University of Technology, Hamburg, Germany  
Email: Bauch@tuhh.de

**Abstract**—We provide two analytical tools to model the inactivation decoding process of LT codes. First, a model is presented which derives the expected number of inactivations occurring in the decoding process of an LT code. This analysis is then extended allowing the derivation of the distribution of the number of inactivations. The accuracy of the method is verified by Monte Carlo simulations. The proposed analysis opens the door to the design of LT codes optimized for inactivation decoding.

## I. INTRODUCTION

Fountain codes [1] are a class of erasure correcting codes which can potentially generate an unbounded number of encoded symbols. This feature makes them very useful when the erasure probability of the communication channel is not known at the transmitter. Fountain codes are also a very efficient solution for reliable multicast/broadcast transmissions. The first class of practical fountain codes, Luby transform (LT) codes, was introduced in [2] together with an efficient belief propagation (BP) erasure decoding algorithm exploiting a bipartite graph representation of the codes. BP decoding of LT codes performs remarkably well for long source blocks but its performance degrades remarkably when applied to moderate and short lengths [3]. Raptor codes were introduced in [4] as a modification of LT codes. They consist of a serial concatenation of an LT code with an outer (fixed-rate) code that is normally chosen to be a high rate erasure correcting code.

LT and Raptor codes are often designed and analyzed assuming BP decoding and very large source blocks. However, frequently in practice moderate source block sizes are used. For example, the Raptor codes standardized in [5] and [6] assume a source block length which ranges from 1024 to 8192 source symbols. The performance of LT codes under BP decoding in the finite length regime was analyzed in [3], [7], [8]. For short to moderate-length source blocks an efficient maximum likelihood (ML) decoding algorithm exists which has a manageable complexity and it is actually

widely used in practice [9], [5]. This algorithm is commonly referred to as *inactivation decoding*. The erasure correcting performance of LT codes under ML decoding was studied in [10]. In [11] a simple model of inactivation decoding for LT codes was presented which provides an approximation of the number of inactivations needed for decoding. The method in [11] is not always accurate. In this paper we present an accurate finite length analysis of LT codes under inactivation decoding following the approach used in [7] for the analysis of BP decoding. The analysis we present is not only able to determine the average number of inactivations, but also to provide the distribution of the number of inactivations. This is important for short LT codes, since in this regime substantial deviations from the average number of inactivations can be expected.<sup>1</sup>

The paper is organized as follows. In Section II we introduce inactivation decoding of LT codes. Section III introduces a first order analysis of inactivation decoding of LT codes which is able to provide the expected number of inactivations needed for decoding. In Section IV we outline an analysis that provides the distribution of the number of inactivations. Finally we present the conclusions to our work in Section V.

## II. INACTIVATION DECODING OF LT CODES

A binary LT code is considered with  $k$  input symbols  $\mathbf{v} = (v_1, v_2, \dots, v_k)$ . The output degree distribution is denoted by  $\Omega = \{\Omega_1, \Omega_2, \Omega_3, \dots, \Omega_{d_{\max}}\}$ , where  $d_{\max} \leq k$  is the maximum output degree. We assume the receiver has collected  $m = k + \delta$  output symbols,  $\mathbf{c} = (c_1, c_2, \dots, c_m)$ . The parameter  $\delta$  is usually referred to as absolute receiver overhead. We denote by  $\epsilon = m/k - 1$  the relative receiver overhead. Decoding consists of solving the linear system of equations

$$\mathbf{c} = \mathbf{v}\mathbf{G}^T$$

where  $\mathbf{G}$  is the  $m \times k$  binary matrix which defines the relation between the input and the output symbols. For LT codes the matrix  $\mathbf{G}$  is usually sparse. This sparsity can be exploited to perform ML decoding in an efficient way [9], [13]–[15] through a decoding algorithm that is commonly

<sup>1</sup>In [12] a finite length analysis of batched sparse codes was introduced, that can also be applied to LT codes. This analysis provides the expected number of inactivations, like the analysis in Section III in this paper.

This work has been accepted for publication at the 53rd Annual Allerton Conference on Communication, Control, and Computing, 2015.

©2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting /republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

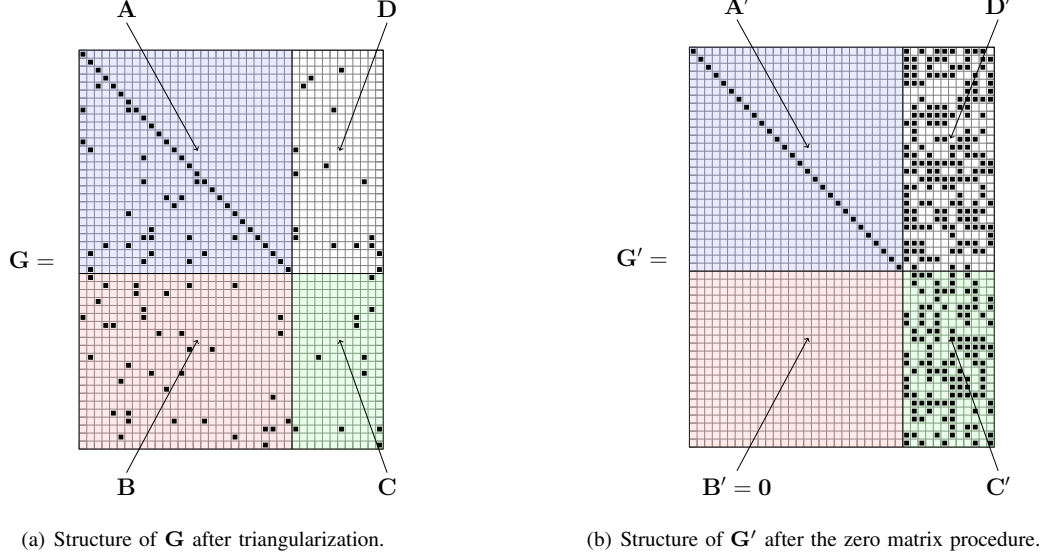


Fig. 1. Triangularization and zero matrix procedure steps of inactivation decoding.

referred to as *inactivation decoding* and that consists of the following steps:

- 1) *Triangularization*.  $\mathbf{G}^T$  is put in an approximate lower triangular form. At the end of this process we are left with lower triangular matrix  $\mathbf{A}$  and matrices  $\mathbf{B}$ ,  $\mathbf{C}$ , and  $\mathbf{D}$  which are sparse as shown in Fig. 1(a). The columns of  $\mathbf{G}$  corresponding to matrices  $\mathbf{C}$  and  $\mathbf{D}$  are usually referred to as inactive columns. This process consists of column and row permutations.
- 2) *Zero matrix procedure*. The matrix  $\mathbf{A}$  is put in a diagonal form and matrix  $\mathbf{B}$  is zeroed out through row sums. As a consequence matrices  $\mathbf{C}$  and  $\mathbf{D}$  tend to become dense. Fig. 1(b) shows The structure of  $\mathbf{G}'$  at the end of this procedure.
- 3) *Gaussian elimination (GE)*. GE is applied to solve the systems of equations  $\tilde{\mathbf{c}} = \tilde{\mathbf{v}}[\mathbf{C}']^T$ , where the symbols in  $\tilde{\mathbf{v}}$  are called *inactive variables* (associated with the columns of the matrix  $\mathbf{C}'$  in Fig. 1(b)) and  $\tilde{\mathbf{c}}$  are known terms associated with the rows of the matrix  $\mathbf{C}'$  in Fig. 1(b). This step drives the cost of inactivation decoding since its complexity is cubic in the number of inactivations.
- 4) *Back-substitution*. Once the values of the inactive variables have been determined, back-substitution is applied to compute the values of the remaining variables in  $\mathbf{v}$ .

Decoding succeeds only if the rank of the matrix  $\mathbf{C}'$  equals the number of inactive variables.

In this paper we focus on the triangularization step since it is the one that determines the number of inactivations. In the remainder of the paper we will use a bipartite graph representation of the LT code. At the left hand side of the bipartite graph we will show the source symbols and at the right hand side output symbols. The adjacency matrix of the bipartite graph is given by matrix  $\mathbf{G}$ , where source symbols

correspond to columns and output symbols to rows of  $\mathbf{G}$ . Due to the one-to-one correspondence between nodes and symbols, we will use both names interchangeably.

The triangularization step can be represented by an iterative pruning of the bipartite graph of the LT code. At each step, a reduced graph is obtained as the sub-graph of the original LT code graph involving only a subset of the input symbols (that we call *active* input symbols) and their neighbors. We will use the term *reduced* degree of a node (symbol) to refer to the degree of a node (symbol) in the reduced graph. Hence, the reduced degree of a node (symbol) is less or equal to its (original) degree. We will use the notation  $\deg(c) = d$  to refer to the (original) degree of an output symbol. Let us now introduce some additional definitions that will be used to model the triangularization step.

**Definition 1 (Ripple).** We define the ripple as the set of output symbols of reduced degree 1 and we denote it by  $\mathcal{R}$ .

The cardinality of the ripple will be denoted by  $r$  and the corresponding random variable as  $R$ .

**Definition 2 (Cloud).** We define the cloud as the set of output symbols of reduced degree  $d \geq 2$  and we denote it by  $\mathcal{C}$ .

The cardinality of the cloud will be denoted by  $c$  and the corresponding random variable as  $C$ .

Figure 2 shows an example of bipartite graph with for an LT code with 4 source symbols and 4 output symbols. In the graph we can see how all 4 source symbols are active. If we now look at the output symbols we can see how the ripple and the cloud are composed of two elements each,  $\mathcal{R} = \{c_1, c_4\}$  and  $\mathcal{C} = \{c_2, c_3\}$ . Before triangularization starts all source symbols are marked as active. At every step of the process, triangularization marks one active source symbol as *resolvable* or *inactive* and the

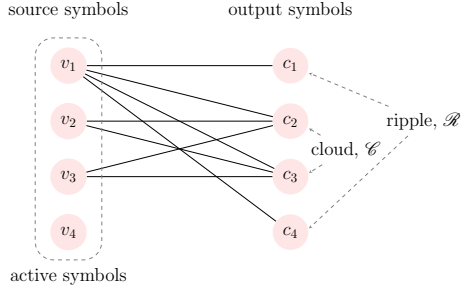


Fig. 2. Example of bipartite graph for an LT code. Source symbols are represented at the left hand side, output symbols at the right hand side.

symbol leaves the reduced graph. Therefore, the reduced graph will initially correspond to the bipartite graph of the LT code. After  $k$  steps the reduced graph will correspond to an empty graph. In the following, for all the definitions provided above we will add a temporal dimension through the subscript  $u$  that corresponds to the number of active input symbols in the graph. As we will see next, since at each step of the triangularization procedure the number of active input symbols decreases by 1,  $u$  decreases as the algorithm progresses. More specifically, the triangularization will start with  $u = k$  active symbols and it will end after  $k$  steps with  $u = 0$ . The following algorithm describes the triangularization procedure at step  $u$  (i.e., in the transition to from  $u$  to  $u - 1$  active symbols):

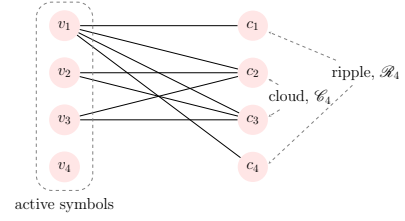
**Algorithm 1** (Triangularization with random inactivations).

- If the ripple  $\mathcal{R}_u$  is not empty ( $r_u > 0$ )  
*The decoder selects an output symbol  $c \in \mathcal{R}_u$  uniformly at random. The only neighbor of  $c$ , i.e. the input symbol  $v$ , is marked as resolvable and leaves the reduced graph. The edges attached to  $v$  are removed.*
- If the ripple  $\mathcal{R}_u$  is empty ( $r_u = 0$ )  
*An inactivation takes place. One the input active symbols,  $v$ , is chosen uniformly at random.<sup>2</sup> This input symbol is marked as inactive and leaves the reduced graph. The edges attached to  $v$  are removed.*

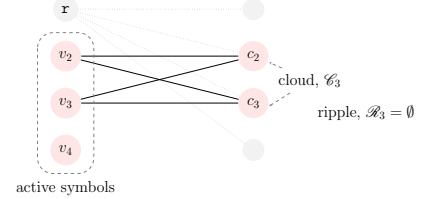
At the end of the procedure, the source symbols which are marked as resolvable correspond to the columns of matrices **A** and **B** in Figure 1(a). Similarly, the source symbols marked as inactive correspond to the columns of matrices **C** and **D**.

**Example 1.** In order to illustrate Algorithm 1 in Figure 3 we provide an example for an LT code with  $k = 4$  source symbols and  $m = 4$  output symbols.

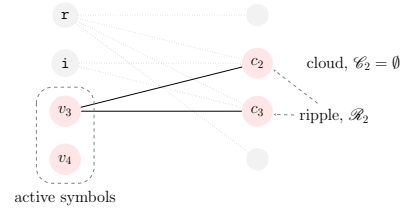
<sup>2</sup>This is certainly neither the only possible inactivation strategy nor the one leading to the least number of inactivations. However, this strategy makes the analysis trackable. For an overview of the different inactivation strategies we refer the reader to [16].



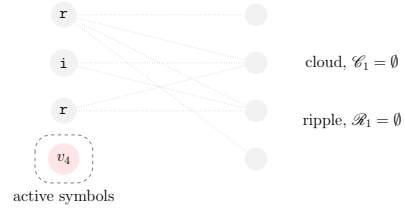
(a) LT code graph example,  $u = 4$



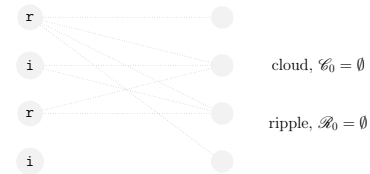
(b) LT code graph example,  $u = 3$



(c) LT code graph example,  $u = 2$



(d) LT code graph example,  $u = 1$



(e) LT code graph example,  $u = 0$

Fig. 3. Example of triangularization process for an LT code.

- 1) *Transition from  $u = 4$  to  $u = 3$ . In Figure 3(a) we can observe how at the initial step  $u = 4$ , the ripple is not empty,  $r_4 = 2$ . Hence, in the transition to  $u = 3$  one of the source symbols ( $v_1$ ) is marked as resolvable, it leaves the graph and all its attached edges are removed, see Figure 3(b). The nodes  $c_1$  and  $c_4$  leave the graph since their reduced degree becomes zero.*
- 2) *Transition from  $u = 3$  to  $u = 2$ . In Figure 3(b) we can see how the ripple is empty,  $r_3 = 0$ . Therefore, in the transition to  $u = 2$  an inactivation takes place. Node  $v_2$  is chosen at random and becomes inactive. All edges attached to  $v_2$  are removed from the graph. As a consequence nodes  $c_2$  and  $c_3$  leave the cloud  $\mathcal{C}_3$  and enter the ripple  $\mathcal{R}_2$ , as it can be seen in Figure 3(c).*
- 3) *Transition from  $u = 2$  to  $u = 1$ . We can see in Figure 3(c) how the ripple is not empty, in fact,  $r_2 = 2$ . Source symbol  $v_3$  is marked as resolvable and all its attached edges are removed. Nodes  $c_2$  and  $c_3$  leave the graph because their reduced degree becomes zero (see Figure 3(d)).*
- 4) *Transition from  $u = 1$  to  $u = 0$ . In Figure 3(d) we can see how the ripple and cloud are now empty. Hence, an inactivation takes place: node  $v_4$  is marked as inactive and the triangularization procedure ends.*

Let us remark that the definition of the triangularization procedure given in Algorithm 1 has the peculiarity that it never stops before  $k$  steps regardless of the properties of the bipartite graph. For example, if some of the input symbols have no edge to any output symbols they are simply marked as inactive at some step, as it happened in our example with node  $v_4$ .

### III. FIRST ORDER ANALYSIS FOR INACTIVATION DECODING

We follow the model introduced in [3], [7], [8] for BP decoding of LT codes. In our work we show how this approach can also be used to model inactivation decoding of LT codes. The decoder is modelled as a finite state machine with state

$$S_u := (C_u, R_u).$$

In this section we derive a recursion which allows to obtain  $\Pr\{S_{u-1} = (c_{u-1}, r_{u-1})\}$  as a function of  $\Pr\{S_u = (c_u, r_u)\}$ .

We shall first analyze how the ripple and cloud change in the transition from  $u$  to  $u - 1$  active source symbols. Since in the transition exactly one active source symbol is marked as either resolvable or inactive and all its attached edges are removed, the degree of some of the output symbols may be reduced. Consequently, some of the cloud symbols may enter the ripple and some of the ripple symbols may become of reduced degree zero and leave the reduced graph. Let us first focus on the symbols that leave the cloud in the transition given that  $S_u = (c_u, r_u)$ . Since in an LT code the output symbols neighbors are selected uniformly at random, the number of cloud symbol which leave  $\mathcal{C}_u$  and enter  $\mathcal{R}_{u-1}$

is binomially distributed with parameters  $c_u$  and  $p_u$ , being  $p_u$  the probability of a symbol leaving  $\mathcal{C}_u$  to enter  $\mathcal{R}_{u-1}$ ,

$$\begin{aligned} p_u &:= \Pr\{c \in \mathcal{R}_{u-1} | c \in \mathcal{C}_u\} \\ &= \frac{\Pr\{c \in \mathcal{R}_{u-1}, c \in \mathcal{C}_u\}}{\Pr\{c \in \mathcal{C}_u\}}. \end{aligned} \quad (1)$$

We are first interesting in evaluating  $\Pr\{c \in \mathcal{R}_{u-1}, c \in \mathcal{C}_u | \deg(c) = d\}$  which corresponds to the probability that one of the edges of a degree- $d$  output symbol  $c$  connected to the symbol being marked as inactive or resolvable at the transition, one edge to one of the  $u - 1$  active symbols after the transition and the remaining  $d - 2$  edges connected to the  $k - u$  not active input symbols (inactive or resolvable). In other words, the symbol must have *reduced degree 2 before the transition and reduced degree 1 after the transition*.

**Proposition 1.** *The probability that a symbol  $c$  belongs to the cloud at step  $u$  and enters the ripple at step  $u - 1$ , given its original degree  $d$  is given by*

$$\begin{aligned} \Pr\{c \in \mathcal{R}_{u-1}, c \in \mathcal{C}_u | \deg(c) = d\} = \\ \frac{d}{k} (d - 1) \frac{u - 1}{k - 1} \frac{\binom{k-u}{d-2}}{\binom{k-2}{d-2}} \end{aligned} \quad (2)$$

for  $d \geq 2$ , while  $\Pr\{c \in \mathcal{R}_{u-1}, c \in \mathcal{C}_u | \deg(c) = d\} = 0$  for  $d < 2$ .

*Proof:* The probability of an edge connecting to the the symbol being marked as inactive or resolvable at the transition is  $1/k$ , and there are  $d$  distinct choices for the edge connected to it. This accounts for the term  $d/k$  in (2). Moreover, there are  $d - 1$  choices for the edge going to the  $u - 1$  active symbols after the transition, while the probability of an edge being connected to the set of  $u - 1$  active symbols is  $(u - 1)/(k - 1)$ . This is reflected in the term  $(d - 1)(u - 1)/(k - 1)$  in (2). The probability of having exactly  $d - 2$  edges going to the  $k - u$  not active input symbols is finally

$$\frac{\binom{k-u}{d-2}}{\binom{k-2}{d-2}}.$$

By removing the conditioning on  $d$  in (2) we obtain

$$\begin{aligned} \Pr\{c \in \mathcal{R}_{u-1}, c \in \mathcal{C}_u\} = \\ \sum_{d=2}^{d_{\max}} \Omega_d \frac{d}{k} (d - 1) \frac{u - 1}{k - 1} \frac{\binom{k-u}{d-2}}{\binom{k-2}{d-2}}. \end{aligned} \quad (3)$$

The denominator of (1) is given by the probability that the randomly chosen output symbol  $c$  is in the cloud when  $u$  input symbols are still active. This is equivalent to the probability of not being in the ripple or having reduced degree zero (all edges are going to symbols marked as inactive or resolvable) as provided by the following Proposition.

**Proposition 2.** *The probability that the randomly chosen output symbol  $c$  is in the cloud when  $u$  input symbols are*

still active is

$$\Pr\{c \in \mathcal{C}_u\} = 1 - \sum_{d=1}^{d_{\max}} \Omega_d \left[ u \frac{\binom{k-u}{d-1}}{\binom{k}{d}} + \frac{\binom{k-u}{d}}{\binom{k}{d}} \right]. \quad (4)$$

*Proof:* The probability of  $c$  not being in the cloud is given by the probability of  $c$  being in the ripple or having reduced degree 0. Being the two events mutually exclusive, we can compute such probability as the sum of two probabilities, the probability of  $c$  having reduced degree 1 (i.e., of  $c$  being in the ripple) and the probability of  $c$  having reduced degree 0. Let us focus on the first of the two probabilities. Assuming the degree of  $c$  being  $d$ , the probability that  $c$  has reduced degree 1 equals the probability of  $c$  having one neighbor among the  $u$  active source symbols and  $d-1$  neighbors among the  $k-u$  non-active ones. This is given by

$$d \frac{u}{k} \frac{\binom{k-u}{d-1}}{\binom{k-1}{d-1}},$$

that corresponds to the first term in (4). The probability of  $c$  having reduced degree 0 is the probability of having all  $d$  neighbors in the  $k-u$  non-active symbols, leading to the term

$$\frac{\binom{k-u}{d}}{\binom{k}{d}}$$

in (4).  $\blacksquare$

The probability  $p_u$  can be finally obtained through (1) making use of (3) and of (4).

We are interested now in analyzing the number  $a_u$  of output symbols leaving the ripple during the transition at step  $u$ . We denote by  $A_u$  the random variable associated with  $a_u$ . We distinguish two cases. In the first case, the ripple is not empty. In this case no inactivation takes place. Hence, an output symbol is chosen at random and removed from the ripple. Any other output symbol in the ripple which is connected to the chosen input symbol leaves the ripple during the transition. Hence, for  $r_u > 0$  we have

$$\Pr\{A_u = a_u | R_u = r_u\} = \binom{r_u - 1}{a_u - 1} \left(\frac{1}{u}\right)^{a_u - 1} \left(1 - \frac{1}{u}\right)^{r_u - a_u}$$

with  $1 \leq a_u \leq r_u$ . In the second case, the ripple is empty ( $r_u = 0$ ). Since no output symbols can leave the ripple, we have  $\Pr\{A_u = 0 | R_u = 0\} = 1$ . Now we are in the position to derive the transition probability

$$\Pr\{S_{u-1} = (c_{u-1}, r_{u-1}) | S_u = (c_u, r_u)\}.$$

Introducing  $b_u := c_u - c_{u-1}$  and observing that  $a_u - b_u =$

$r_u - r_{u-1}$  we have

$$\Pr\{S_{u-1} = (c_u - b_u, r_u - a_u + b_u) | S_u = (c_u, r_u)\} = \binom{c_u}{b_u} p_u^{b_u} (1 - p_u)^{c_u - b_u} \binom{r_u - 1}{a_u - 1} \times \left(\frac{1}{u}\right)^{a_u - 1} \left(1 - \frac{1}{u}\right)^{r_u - a_u} \quad (5)$$

for  $r_u > 0$ , while

$$\Pr\{S_{u-1} = (c_u - b_u, b_u) | S_u = (c_u, 0)\} = \binom{c_u}{b_u} p_u^{b_u} (1 - p_u)^{c_u - b_u}. \quad (6)$$

Therefore, the probability of the decoder being in state  $S_{u-1} = (c_{u-1}, r_{u-1})$  can be computed recursively via (5), (6) with the initial condition

$$\Pr\{S_k = (c_k, r_k)\} = \binom{m}{r_k} \Omega_1^{r_k} (1 - \Omega_1)^{c_k}$$

for all non-negative  $c_k, r_k$  such that  $c_k + r_k = m$  where  $m$  is the number of output symbols. Denoting by  $N$  the random variable modelling the cumulative number of inactivations after  $k$  steps, its expected value is finally given by

$$\mathbb{E}[N] = \sum_{u=1}^k \sum_{c_u} \Pr\{S_u = (c_u, 0)\}. \quad (7)$$

Figure 4 shows the expected number of inactivations for a  $k = 1000$  LT code with output degree distribution [5], [6]

$$\begin{aligned} \Omega(x) &:= \sum_d \Omega_d x^d \\ &= 0.0098x + 0.4590x^2 + 0.2110x^3 + 0.1134x^4 + \\ &\quad 0.1113x^{10} + 0.0799x^{11} + 0.0156x^{40}. \end{aligned}$$

The chart reports the average number of inactivations obtained through both Monte Carlo simulation and by (7), and shows a tight match between the analysis and the simulation results.

#### IV. DISTRIBUTION OF THE NUMBER OF INACTIVATIONS

The analysis presented in Section III is able to capture the expected number of inactivations. We shall see next that the model can be easily modified to obtain the distribution of the number of inactivations. To do so, we first need to extend the finite state machine by including in the state definition the number of inactive input symbols, i.e.,

$$S_u = (C_u, R_u, N_u)$$

with  $N_u$  being the random variable modelling the number of inactivations at step  $u$ . Again, we proceed by deriving a recursion which allows deriving  $\Pr\{S_{u-1} = (c_{u-1}, r_{u-1}, n_{u-1})\}$  as a function of  $\Pr\{S_u = (c_u, r_u, n_u)\}$ . We will look first at the transition from  $u$  to  $u-1$  active symbols when  $r_u \geq 1$ , that is, when no inactivation takes

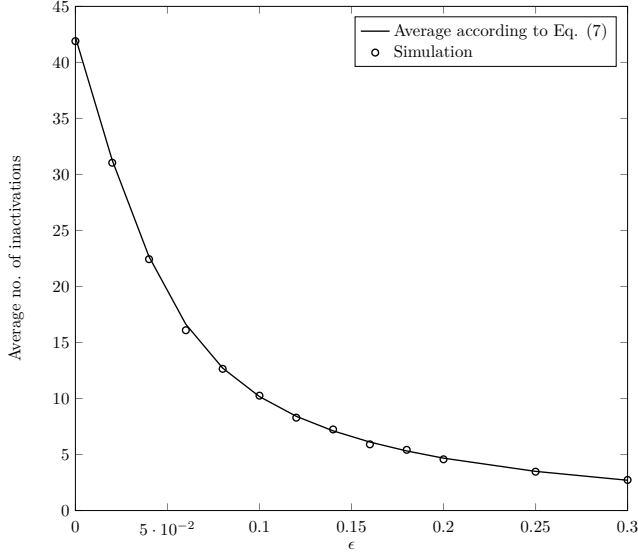


Fig. 4. Average number of inactivations vs. relative overhead for an LT code with degree distribution  $\Omega(x) = 0.0098x + 0.4590x^2 + 0.2110x^3 + 0.1134x^4 + 0.1113x^{10} + 0.0799x^{11} + 0.0156x^{40}$ . The source block size is  $k = 1000$ .

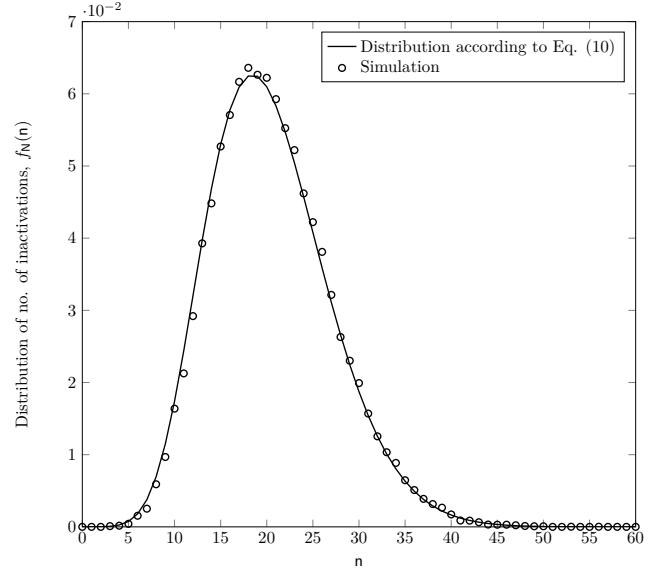


Fig. 5. Distribution of the number of inactivations for an LT code with degree distribution  $\Omega(x) = 0.0098x + 0.4600x^2 + 0.2110x^3 + 0.1134x^4 + 0.1110x^{10} + 0.0800x^{11} + 0.0156x^{40}$ . The source block size is  $k = 300$ .

place. In this case the number of inactivations does not increase and we have  $n_{u-1} = n_u$ . We have thus

$$\Pr\{S_{u-1} = (c_u - b_u, r_u - a_u + b_u, n_u) | S_u = (c_u, r_u, n_u)\} = \binom{c_u}{b_u} p_u^{b_u} (1 - p_u)^{c_u - b_u} \times \binom{r_u - 1}{a_u - 1} \left(\frac{1}{u}\right)^{a_u - 1} \left(1 - \frac{1}{u}\right)^{r_u - a_u}. \quad (8)$$

Let us now look at the transition from  $u$  to  $u - 1$  active symbols when  $r_u = 0$ , that is, when an inactivation takes place. In this case the number of inactivations increases by one yielding

$$\Pr\{S_{u-1} = (c_u - b_u, b_u, n_u + 1) | S_u = (c_u, 0, n_u)\} = \binom{c_u}{b_u} p_u^{b_u} (1 - p_u)^{c_u - b_u}. \quad (9)$$

The probability of the decoder being in state  $S_{u-1} = (c_{u-1}, r_{u-1}, n_{u-1})$  can be computed recursively via (8), (9) with the initial condition

$$\Pr\{S_k = (c_k, r_k, n_k)\} = \binom{m}{r} \Omega_1^r (1 - \Omega_1)^{c_k}$$

for all non-negative  $c_k, r_k$  such that  $c_k + r_k = m$  and  $n_k = 0$ .

The distribution of the number of inactivations needed to complete the decoding process is finally given by

$$f_N(n) = \sum_{c_0} \sum_{r_0} \Pr\{S_0 = (c_0, r_0, n)\}. \quad (10)$$

From (10) we may obtain the cumulative distribution  $F_N(n)$  which would give the probability of performing at most  $n$

inactivations during the decoding process. The cumulative distribution of the number of inactivations has practical implications. Assume the fountain decoder runs on a platform with limited computational capability. For example, the decoder may be able to perform a maximum number of inactivations (e.g., due to the complexity associated with the third step of the algorithm outlined in Section II, which grows cubically with the number of inactivations). Suppose the maximum number of inactivations that the decoder can handle is  $n^*$ . For such a decoder, the probability of decoding failure will be lower bounded by  $F_N(n^*)$ .<sup>3</sup>

Fig. 5 shows the distribution of the number of inactions, for an LT code with degree  $\Omega(x)$  from Section III and source block size  $k = 300$ . The distribution of inactivations has been obtained through both Monte Carlo simulation and by (10), showing again a tight match between the analysis and the simulation results.

## V. CONCLUSIONS AND DISCUSSION

We have introduced a novel analysis of inactivation decoding of LT codes. A first analysis provides the expected number of inactivations needed to by an LT decoder. Furthermore, we have presented an extended analytical approach which is able to provide the distribution of the number of inactivations. The later analysis is especially important for the design of LT codes under inactivation decoding since it captures the deviations of the number of inactivations with respect to the average.

<sup>3</sup>The probability of decoding failure is actually higher than  $F_N(n^*)$  since the system of equations to be solved in the Gaussian elimination (GE) step of inactivation decoding might be rank deficient.

## REFERENCES

- [1] J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to reliable distribution of bulk data," *IEEE J. Select. Areas Commun.*, vol. 20, no. 8, pp. 1528–1540, Oct. 2002.
- [2] M. Luby, "LT codes," in *Proc. of the 43rd Annual IEEE Symp. on Foundations of Computer Science*, Vancouver, Canada, Nov. 2002, pp. 271–282.
- [3] G. Maatouk and A. Shokrollahi, "Analysis of the second moment of the LT decoder," *IEEE Trans. Inform. Theory*, vol. 58, no. 5, pp. 2558–2569, May 2012.
- [4] M. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.
- [5] 3GPP TS 26.346 V11.1.0, "Technical specification group services and system aspects; multimedia broadcast/multicast service; protocols and codecs," Jun. 2012.
- [6] M. Luby, A. Shokrollahi, M. Watson, and T. Stockhammer, "RFC 5053: Raptor forward error correction scheme: Scheme for object delivery," Tech. rep., IETF, Tech. Rep., Oct. 2007.
- [7] R. Karp, M. Luby, and A. Shokrollahi, "Finite length analysis of LT codes," in *Proc. 2004 IEEE International Symp. on Inf. Theory*, Chicago, Illinois, US, Jun. 2004.
- [8] A. Shokrollahi, "Theory and applications of Raptor codes," *Math-know*, vol. 3, pp. 59–89, 2009.
- [9] M. Shokrollahi, S. Lassen, and R. Karp, "Systems and processes for decoding chain reaction codes through inactivation," Feb. 2005, US Patent 6,856,263.
- [10] B. Schotsch, G. Garrammone, and P. Vary, "Analysis of lt codes over finite fields under optimal erasure decoding," *IEEE Commun. Lett.*, vol. 17, no. 9, pp. 1826–1829, Sep. 2013.
- [11] F. Lázaro Blasco, G. Liva, and G. Bauch, "LT code design for inactivation decoding," in *Proc. 2014 IEEE Information Theory Workshop*, Hobart, Australia, Nov. 2014, pp. 441–445.
- [12] T.-C. Ng and S. Yang, "Finite-length analysis of BATS codes," in *Network Coding (NetCod), 2013 International Symposium on*, Jun. 2013, pp. 1–6.
- [13] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 638–656, Feb. 2001.
- [14] H. Pishro-Nik and F. Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 50, no. 3, pp. 439–454, Mar. 2004.
- [15] D. Burshtein and G. Miller, "An efficient maximum likelihood decoding of LDPC codes over the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 50, no. 11, pp. 2837–2844, Nov. 2004.
- [16] E. Paolini, G. Liva, B. Matuz, and M. Chiani, "Maximum likelihood erasure decoding of LDPC codes: pivoting algorithms and code design," *IEEE Trans. Commun.*, vol. 60, no. 11, pp. 3209–3220, Nov. 2012.